

A Reduced Latency List Decoding Algorithm for Polar Codes

Jun Lin, Chenrong Xiong and Zhiyuan Yan

Department of Electrical and Computer Engineering, Lehigh University, PA, USA

Email: {jul311, chx310, yan}@lehigh.edu

Abstract—The cyclic redundancy check (CRC) aided successive cancellation list (SCL) decoding algorithm has better error performance than the successive cancellation (SC) decoding algorithm for short or moderate polar codes. However, the CRC aided SCL (CA-SCL) decoding algorithm still suffer from long decoding latency. In this paper, a reduced latency list decoding (RLLD) algorithm for polar codes is proposed. For the proposed RLLD algorithm, all rate-0 nodes and part of rate-1 nodes are decoded instantly without traversing the corresponding subtree. A list maximum-likelihood decoding (LMLD) algorithm is proposed to decode the maximum likelihood (ML) nodes and the remaining rate-1 nodes. Moreover, a simplified LMLD (SLMLD) algorithm is also proposed to reduce the computational complexity of the LMLD algorithm. Suppose a partial parallel list decoder architecture with list size $L = 4$ is used, for an (8192, 4096) polar code, the proposed RLLD algorithm can reduce the number of decoding clock cycles and decoding latency by 6.97 and 6.77 times, respectively.

I. INTRODUCTION

Polar codes [1] are a significant breakthrough in coding theory, since it is proved that polar codes can achieve the channel capacity of binary-input symmetric memoryless channels in [1] and any discrete or continuous channel in [2]. Polar codes can be efficiently decoded by the low-complexity successive cancellation (SC) decoding algorithm [1] with complexity of $O(N \log N)$, where N is the block length.

Lots of efforts [3], [4] have already been devoted to improve the error-correction performance of polar codes with short or moderate lengths. An successive cancellation list (SCL) decoding algorithm was recently proposed in [3], performs better than the SC decoding algorithm and performs almost the same as a maximum-likelihood (ML) decoder [3]. In [4], the cyclic redundancy check (CRC) is used to pick the output codeword from L candidates, where L is the list size. The CRC-aided SCL decoding algorithm performs much better than the SCL decoding algorithm at the expense of negligible loss in code rate. For example, it was shown in [4] that the CRC-aided SCL decoding algorithm outperforms the SC decoding algorithm by more than 1 dB when the bit error rate (BER) is on the order of 10^{-5} for a polar code of length 2048.

Many research efforts [5]–[9] have been devoted to the reduction of the decoding latency of the SC decoding algorithm. The simplified successive cancellation (SSC) and the ML-SSC decoding algorithms were proposed in [5] and [7], respectively. Both SSC and ML-SSC decoding algorithms can reduce the decoding latency of a SC decoder significantly.

However, the reduced latency list decoding algorithm has been rarely discussed in open literature.

In this paper, the algorithms that reduce the latency of list polar decoders are investigated. The main contributions are shown as follows.

- 1) A reduced latency list decoding (RLLD) algorithm over LLR domain for polar codes is proposed. The proposed RLLD algorithm deals with rate-0 nodes and part of rate-1 nodes in the same way as the SSC decoding algorithm.
- 2) A list ML decoding (LMLD) algorithm is proposed to decode the ML and remaining rate-1 nodes. For the list size $L \leq 8$, a hardware friendly simplified LMLD (SLMLD) algorithm is also proposed.
- 3) For list size $L = 4$, an efficient hardware architecture for the proposed SLMLD algorithm is presented. Under a TSMC 90nm technology, at the cost of 1.07 million standard NAND gates, the proposed architecture can achieve a frequency of 400MHz with 4 stage of pipelines.
- 4) For a partial parallel decoder architecture with $L = 4$, it is shown that the RLLD with the SLMLD algorithms can reduce the decoding cycles and latency by 6.97 and 6.77 times, respectively.

II. PRELIMINARIES

A. Polar codes encoding

The generation matrix of a polar code is an $N \times N$ matrix $G = B_N F^{\otimes n}$, where $N = 2^n$, B_N is the bit reversal permutation matrix, and $F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Here $\otimes n$ denotes the n th Kronecker power and $F^{\otimes n} = F \otimes F^{\otimes (n-1)}$. Let $u_0^{N-1} = (u_0, u_1, \dots, u_{N-1})$ denote the data bit sequence and $x_0^{N-1} = (x_0, x_1, \dots, x_{N-1})$ the corresponding encoded bit sequence, then $x_0^{N-1} = u_0^{N-1} G$. The indices of the encoding bit sequence u_0^{N-1} are divided into two sets: the information bits set \mathcal{A} contains K indices and the frozen bits set \mathcal{A}^c contains $N - K$ indices. $u_{\mathcal{A}}$ are the information bits whose indices all come from \mathcal{A} . $u_{\mathcal{A}^c}$ are the frozen bits whose indices from \mathcal{A}^c . The encoding graph of a polar code with $N = 8$ is shown in Fig. 1.

B. SSC and ML-SSC Decoding Algorithms

A polar code of length $N = 2^n$ can also be represented by a full binary tree of depth n [5], where each node of the tree is associated with a constituent code. The binary tree representation of an (8, 3) polar code is shown in Fig. 2, where the black and white leaf nodes correspond to information

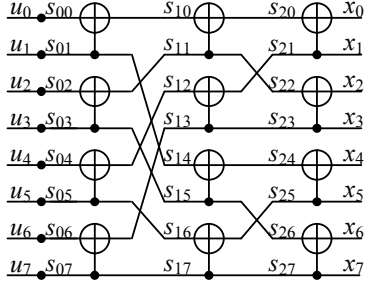


Fig. 1. Polar encoder with $N = 8$

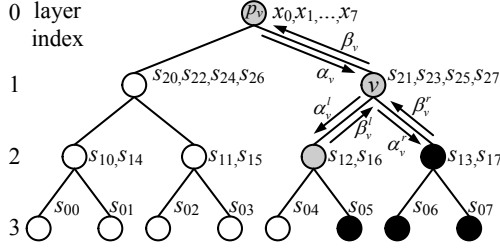


Fig. 2. Binary tree representation of a (8, 3) polar code

and frozen bits, respectively. In order to show the connection between the tree representation and the direct encoding graph in Fig. 1, the constituent code associated with each tree node is also shown in Fig. 2. There are three types of nodes in a binary tree representation of a polar code: rate-0, rate-1 and arbitrary rate nodes. The leaf nodes of a rate-0 and rate-1 nodes are associated with only frozen and information bits, respectively. The leaf nodes of an arbitrary rate node are associated with both information and frozen bits. For example, the rate-0, rate-1 and arbitrary rate nodes in Fig. 2 are represented by circles in white, black and gray, respectively.

The SC decoding algorithm can also be mapped on a binary tree, where each node acts as a decoder for its constituent code. As shown in Fig. 2, the decoder at node v receives a soft information vector α_v and returns its correspondent constituent code β_v . The SC decoding algorithm is initialized by feeding the root node with the channel LLRs, $(L_0, L_1, \dots, L_{N-1})$, where $L_i = \log(\Pr(y_i|x_i = 0)/\Pr(y_i|x_i = 1))$. When an internal node v is activated, it calculates the soft information vector α_v^l sending to its left child, where

$$\alpha_v^l[i] = f(\alpha_v[2i], \alpha_v[2i+1]) \text{ for } 0 \leq i < 2^{n-t}, \quad (1)$$

$f(a, b) = 2 \tanh^{-1}(\tanh(a/2) \tanh(b/2))$, and t is the layer index of the child node. $f(a, b)$ can be approximated as:

$$f(a, b) = \text{sign}(a) \cdot \text{sign}(b) \min(|a|, |b|). \quad (2)$$

Node v then waits until it receives the constituent code β_v^l . The soft information vector

$$\alpha_v^r[i] = \alpha_v[2i](1 - 2\beta_v^l[i]) + \alpha_v[2i+1] \text{ for } 0 \leq i < 2^{n-t}. \quad (3)$$

Once the right child returns its constituent code β_v^r , node v

computes its constituent code β_v as:

$$(\beta_v[2i], \beta_v[2i+1]) = (\beta_v^l[i] \oplus \beta_v^r[i], \beta_v^r[i]), \quad (4)$$

where $0 \leq i < 2^{n-t}$ and \oplus is modulo-2 addition. When a leaf node v is activated, its constituent code β_v is set to 0 if leaf node v is associated with a frozen bit. Otherwise, β_v is calculated from α_v with the threshold detection:

$$\beta_v = \begin{cases} 0 & \alpha_v \geq 0 \\ 1 & \alpha_v < 0 \end{cases} \quad (5)$$

From the root node, all nodes in a tree are activated in a recursive way for the SC decoding. Once β_v for the last leaf node is generated, the codeword x_0^{n-1} can be obtained by combining and propagating β_v up to the root node.

The SSC decoding algorithm in [5] simplifies the decoding of rate-0 and rate-1 nodes. Once a rate-0 node is activated, it immediately returns its constituent code which is an all zero vector. Once a rate-1 node is activated, its constituent code is directly calculated from the received soft information vector with the threshold detection rule shown in Eq. (5). The ML-SSC decoding algorithm [7] simplifies the SSC decoding algorithm further by performing the exhaustive-search ML decoding on some resource constrained arbitrary rate nodes, which are called ML nodes in [7]. For an ML node with layer index t , the associated constituent code is estimated according to:

$$\beta_v = \underset{\mathbf{x} \in \mathcal{C}}{\text{argmax}} \sum_{i=0}^{2^{n-t}-1} (1 - 2\mathbf{x}[i])\alpha_v[i], \quad (6)$$

where \mathcal{C} is the set of possible constituent codes for the ML node. The binary tree representations of the example (8, 3) polar code under SSC and ML-SSC decoding algorithms are shown in Fig. 3 (a) and (b), respectively. It is observed that the SSC decoding algorithm can reduce the number of nodes to be activated. This number is further reduced by applying the ML-SSC decoding algorithm which introduces ML nodes. It is obvious that all the child nodes of a rate-0 and rate-1 node are still rate-0 and rate-1 nodes, respectively. During the reduction of the binary tree, a rate-0 or rate-1 node is kept only if their parent nodes are not a rate-0 or rate-1 node, respectively. For an arbitrary rate node v , let n_v and d_v denote the number of leaf nodes and the number of leaf nodes that correspond to information bits, respectively. In [7], an arbitrary rate node is labeled as an ML node only if its n_v and d_v do not exceed predefined values.

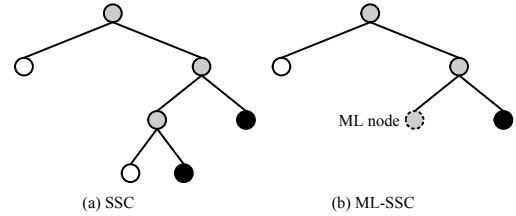


Fig. 3. Binary tree representations of a (8, 3) polar code under SSC and ML-SSC decoding algorithms

C. LLR Based List Decoding Algorithms

In the first several works [3], [10], [11] on list decoding of polar codes, the list decoding algorithm is performed either on probability or logarithmic likelihood (LL) domain. In [12], an LLR based list decoding algorithm is proposed to reduce the message memory requirement and the computational complexity of LL based list decoding algorithm. The LLR based list decoding algorithm employs a novel path metric $\text{PM}_l^{(i)}$, which is computed as:

$$\text{PM}_l^{(i)} = \sum_{k=0}^i m_i |L_n^{(k)}[l]|, \quad (7)$$

where $m_i = 1$ only if $\hat{u}_k[l] = \delta(L_n^{(k)}[l])$ and $\delta(x) = \frac{1}{2}(1 - \text{sign}(x))$ [12]. Otherwise $m_i = 0$. Here $L_n^{(k)}[l] \triangleq \frac{W_n^{(k)}(\mathbf{y}, \hat{u}_0^{k-1}[l]|0)}{W_n^{(k)}(\mathbf{y}, \hat{u}_0^{k-1}[l]|1)}$ and \mathbf{y} is the received channel soft information vector.

III. THE PROPOSED RLLD ALGORITHM

Though existing list decoding algorithms for polar codes can improve the performance of SC decoders significantly. They still suffer from long decoding latency. During the decoding of each information bit, the current decoding paths need to be doubled and at most L most reliable decoding paths are kept, where L is the list size. The extra cycles spent on path pruning increase the number of the overall decoding cycles [10]. In this paper, a reduced latency list decoding (RLLD) algorithm for polar codes is proposed. Let W_v and I_v denote the number of leaf nodes and leaf nodes associated with information bits of a node v in a binary tree, respectively. Let W_T be a predefined threshold value. The general architecture of the proposed RLLD algorithm is shown as follows:

- 1) For a binary tree representation of a polar code, label all the rate-0, rate-1 and ML nodes. For a node v in the tree, let W_v and I_v denote the numbers of leaf nodes and leaf nodes associated with information bits, respectively. For rate-1 nodes, $I_v = W_v$. Moreover, two type of nodes are defined: T_0 and T_1 . T_0 nodes include rate-1 nodes with $I_v > W_T$ and all rate-0 nodes. T_1 nodes include rate-1 nodes with $I_v \leq W_T$ and all ML nodes. For all ML nodes, $W_v \leq W_{ML}$ and $I_v \leq 8$, where W_{ML} is also a predefined threshold value.
- 2) For each decoding path, perform the SC decoding algorithm on the corresponding pruned binary tree, if a T_0 node is activated, the corresponding constituent code is decoded immediately and sent to its parent node. Besides, it is unnecessary to compute the LLR vector sent to a rate-0 node, since the constituent code of a rate-0 node is always a zero vector.
- 3) If a T_1 node is activated, compute 2^{I_v} path metrics for each current decoding path, where each path metric corresponds to the reliability of a possible decoding path. Find at most L most reliable decoding paths and continue their corresponding SC decoding. Since only rate-1 nodes with $I_v < W_T$ are involved in the list

decoding, the choice of W_T should be decided by the numerical simulation.

- 4) Once all T_0 and T_1 nodes have been activated and all the SC decoding procedures on each decoding path are finished, perform cyclic redundancy check (CRC) on the information bits of each candidate codeword. The output codeword is the one that passes the CRC.

In terms of software or hardware implementation, the proposed RLLD algorithm can be performed over L LLR matrices and L bit matrices. For $l = 0, 1, \dots, L-1$ and $t = 1, 2, \dots, n$, let $P_{l,t}$ be a probability message array of 2^{n-t} elements: $P_{l,t}[j]$ stores an LLR message for $j = 0, 1, \dots, 2^{n-t} - 1$. The received channel LLRs are stored in $P_{0,0}$ which has $N = 2^n$ elements. $C_{l,t}$ has a similar structure as $P_{l,t}$: $C_{l,t}[j]$ stores two binary partial sums $C_{l,t}[j][0]$ and $C_{l,t}[j][1]$ for $j = 0, 1, \dots, 2^{n-t} - 1$. Let $r_l = (r_l[n-1], r_l[n-2], \dots, r_l[0])$ be the message updating reference index array for decoding path l . For decoding path l , $r_l[0] \equiv 0$, while all other elements are initialized with 0. When a T_0 or T_1 node v is activated, the computation of the soft information vector sent to node v for decoding path l is shown in Algorithm 1, where t_v is the layer index of node α and P_{l,t_v} is the LLR vector sent to node v . The g function is shown in Eq. (3). If node v is a rate-0 node, as mentioned before, it is unnecessary to compute the received LLR vector. Under this circumstance, t_v is decreased by 1. When a decoding path l needs to be copied to decoding path l' , the lazy copy approach in [10] is applied. Instead of copying LLR matrices, $r_l[I_s - 1], \dots, r_l[1]$ are copied to $r_{l'}[I_s - 1], \dots, r_{l'}[1]$, respectively, while $r_{l'}[n], \dots, r_{l'}[I_s]$ are set to l' .

For decoding path l , during the computation of P_{l,t_v} , LLR arrays, $P_{l,I_s}, \dots, P_{l,t_v}$, need to be updated in serial, where I_s is a pre-computed layer index. For the tree representation of a polar code, suppose all leaf nodes from left to right are indexed from 0 to $N - 1$. Let the indices of the leftmost and rightmost leaf nodes of the subtree of node v be IDX_0 and IDX_1 , respectively. I_s is computed based on IDX_0 as shown in Algorithm 2, where the function dec2bin computes the binary representation of its input and B_{n-1} and B_0 are the most and least significant bits, respectively.

Once the constituent code C_v^l sent from node v for decoding path l is computed, C_v^l is stored in $C_{l,t_v}[k][0]$ for $k = 0, 1, \dots, 2^{n-t_v}$ if node v is the left child of its parent node. Otherwise C_v^l is stored in $C_{l,t_v}[k][1]$ for $k = 0, 1, \dots, 2^{n-t_v}$. If the contents of decoding path l need to be copied to decoding path l' , the partial sums in decoding path l are copied to the corresponding locations in decoding path l' . If node v is the right child of its parent node, then the partial sum computation for path l is performed as shown in Algorithm 3. The input I_e is a layer index and can be obtained by applying Algorithm 2 with IDX_0 and I_s being replaced with IDX_1 and I_e , respectively.

A. LMLD Algorithms

When a T_1 node is activated, the current decoding paths will expand, and at most L most reliable decoding paths are

Algorithm 1: llrComp(l, α)

input : I_s, t_v
output: P_{l,t_v}

```
1 for  $t = I_s$  to  $t_v$  do
2   for  $k = 0$  to  $2^{n-t} - 1$  do
3     if  $t == I_s$  then
4        $b_s = C_{l,t}[k][0]$ 
5        $P_{l,t}[k] =$ 
6          $g(P_{r_l[t-1],t-1}[2k], P_{r_l[t-1],t-1}[2k+1], b_s)$ 
7     else
8        $P_{l,t}[k] =$ 
9          $f(P_{r_l[t-1],t-1}[2k], P_{r_l[t-1],t-1}[2k+1])$ 
```

Algorithm 2:

input : IDX_0
output: I_s

```
1 if  $IDX_0 == 0$  then  $I_s = 0$  else
2    $I_s = n$ 
3    $(B_n, B_{n-1}, \dots, B_0) = \text{dec2bin}(IDX_0)$ 
4   for  $j = 0$  to  $n - 1$  do
5     if  $B_j == 0$  then  $I_s = I_s - 1$  else break
```

kept. In this paper, a list ML decoding (LMLD) algorithm is proposed to find at most L most reliable decoding paths. For a T_1 node v , there are 2^{I_v} candidate output constituent codes since the number of information bits associated with the leaf nodes of a node v is I_v . Therefore, for each decoding path l , the proposed LMLD algorithm computes 2^{I_v} extended path metrics PM_l^j for $j = 0, 1, \dots, 2^{I_v} - 1$ based on the current path metric PM_l . Finding the L most reliable surviving decoding paths is equivalent to find the L most reliable constituent codes among all candidates. Here, several conclusions are made on path metrics and extended path metrics:

- For each decoding path l , the path metric PM_l is initialized with 0. The extended path metrics are computed

Algorithm 3: pSumComp(l, α)

input : I_e, t_v

```
1 for  $t = t_v$  to  $I_e$  do
2   for  $k = 0$  to  $2^{n-t-1} - 1$  do
3     if  $t == I_e$  then
4        $C_{l,t}[2k][0] = C_{l,t-1}[k][0] \oplus C_{l,t-1}[k][1]$ 
5        $C_{l,t}[2k+1][0] = C_{l,t-1}[k][1]$ 
6     else
7        $C_{l,t}[2k][1] = C_{l,t-1}[k][0] \oplus C_{l,t-1}[k][1]$ 
8        $C_{l,t}[2k+1][1] = C_{l,t-1}[k][1]$ 
```

only when a T_1 node is activated.

- For each decoding path l , each extended path metric PM_l^j corresponds to the reliability measure of the associated candidate constituent code $C_{v,l}^j$ sent from node v .
- The extended path metric PM_l^j is computed as shown in Eq. (8), where NM_l^j is called node metric and $NM_l^j = \sum_{k=0}^{2^{I_v}-1} m_k |\alpha_{v,l}[k]|$. $\alpha_{v,l}$ is the LLR vector received by the node v . $m_k = 1$ only if $C_{v,l}^j[k] = \delta(\alpha_{v,l}[k])$, where $\delta(x) = \frac{1}{2}(1 - \text{sign}(x))$. Otherwise, $m_k = 0$. As shown in Eq. (8), for $k = 0, 1, \dots, 2^{I_v} - 1$, if $C_{v,l}^j[k]$ does not equal to the threshold detection based on $\alpha_{v,l}[k]$, then PM_l^j is punished by adding the absolute value of $\alpha_{v,l}[k]$. As a result, the smaller a extended path metric is, the more reliable a corresponding constituent code is.

$$PM_l^j = PM_l + NM_l^j \quad (8)$$

Based on the previous conclusions, the proposed LMLD algorithm finds the L most reliable constituent codes by sorting out the L minimum metrics among $2^{I_v} L$ metrics. Let set $S = \{(l, j)_r | r = 0, 1, \dots, L - 1\}$, where $(l, j)_r$ is the index of a candidate constituent code. Thus, the proposed LMLD algorithm is shown in Eq. (9),

$$S = \underset{j \in [0, 2^{I_v}-1]}{\text{argmin}} -L \underset{l \in [0, L-1]}{PM_l^j}, \quad (9)$$

where $\text{argmin}-L$ finds the associated indices of the L minimum metrics among all input metrics. The current L path metrics are updated with the L minimum extended path metrics.

As shown in Eq. (9), the computational complexity of the proposed LMLD algorithm is exponential to I_v which is the number of leaf nodes associated with information bits for node v . As a result, the maximum value of I_v should be limited for practical implementation of the proposed LMLD algorithm. In this paper, the maximum value of I_v is set to 8. The maximum number of leaf nodes of a ML node is set to $W_{ML} = 16$. In case of W_T is greater than 8, the corresponding rate-1 node is split to several rate-1 nodes with $W_v = 8$. The other generated nodes due to the split are viewed as arbitrary rate nodes. Take a rate-1 node with $W_v = 32$ as an example, the split is shown in Fig. 4, where 4 rate-1 nodes with $W_v = 8$ are generated while the other generated nodes are deemed as arbitrary rate nodes. Besides, W_v for a rate-1 node can only be a power of 2.

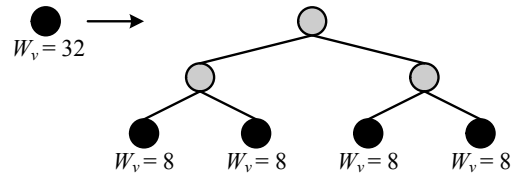


Fig. 4. The tree split of a rate-1 node with $W_v = 32$

B. SLMLD Algorithms

The computational complexity of the proposed LMLD algorithm is still high when I_v is close to 8. In this paper, for $L \leq 8$, a simplified list ML decoding (SLMLD) algorithm suitable for parallel hardware implementation is proposed to reduce the computational complexity of the proposed LMLD algorithm in further. Here, L is assumed to be a power of 2. The proposed SLMLD algorithm shown in Eq. (9) is divided into two major steps:

- 1) For each current decoding path l , find its most reliable L constituent codes based on node metrics. Since only the L most reliable constituent codes are needed at last and at most L constituent codes are from the same decoding path l , it is enough to find the L most reliable constituent codes for a decoding path l .
- 2) Compute the extended path metrics based on survived node metrics from previous step, and find the final L most reliable constituent codes based on these $L \times L$ extended metrics.

Depending on the value of I_v , the first step can be simplified further. If $2^{I_v} \leq L$, nothing needs to be done. If $2^{I_v} = 2L$, the minimum L extended path metrics and their corresponding l and j indices are computed with a bitonic sequence [13] based sorter (BBS) [11], where the BBS first transforms the inputs into a bitonic sequence and then generates L minimum metrics among all inputs. When $2^{I_v} > 2L$, the minimum L node metrics are computed as follows:

- The 2^{I_v} node metrics are divided into L groups as follows:

$$\underbrace{NM_l^0, \dots, NM_l^{q-1}}_{\text{group 1}}, \dots, \underbrace{NM_l^{(L-1)q}, \dots, NM_l^{Lq-1}}_{\text{group } L},$$

where $q = \frac{2^{I_v}}{L}$. The minimum two metrics of each group are then computed.

- Among the resulting $2L$ extended path metrics, the minimum L extended path metrics and their corresponding l and j indices are computed with a BBS.

When list size $L = 2$, for any I_v values, the first step is just finding the minimum two extended path metrics and their corresponding index pairs (l, j) 's.

The second step of the proposed SLMLD algorithm employs the $2L$ - L BBS sorter with $2L$ inputs and L outputs repeatedly to generate L final extended path metrics and their associated path indices. Take $L = 4$ as an example, there are $4L$ extended path metrics: $PM_{l_0}^{j_0}, PM_{l_1}^{j_1}, \dots, PM_{l_{4L-1}}^{j_{4L-1}}$, then $PM_{l_0}^{j_0}, \dots, PM_{l_{2L-1}}^{j_{2L-1}}$ and $PM_{l_{2L}}^{j_{2L}}, \dots, PM_{l_{4L-1}}^{j_{4L-1}}$ are applied to two $2L$ - L BBSs, respectively. Thus, total $2L$ metrics are selected out. Then the $2L$ - L BBS is employed again to generate the final L minimum extended path metrics: $PM_{l'_0}^{j'_0}, PM_{l'_1}^{j'_1}, \dots, PM_{l'_{L-1}}^{j'_{L-1}}$.

C. Simulation Results

For an (8192, 4096) polar code, the frame error rate (FER) performance of the proposed RLLD algorithm are shown in Fig. 5, under the AWGN channel with BPSK modulation.

As shown in Fig. 5, CS_i denotes the CRC aided SC list decoding algorithm [3] with list size $L = i$ over LLR domain, and $RS(i, \omega)$ denotes the proposed RLLD algorithm with the SLMLD algorithm when list size $L = i$ and $W_T = \omega$. For both CS_i and $RS(i, \omega)$ algorithms, 32 information bits are replaced with a 32-bit CRC checksum.

For simplicity, the FER performances of the proposed RLLD algorithm with LMLD (RL) algorithm are not shown in this paper, since the FER performances of the RL algorithm are the same as that of the CS algorithm with the same list size.

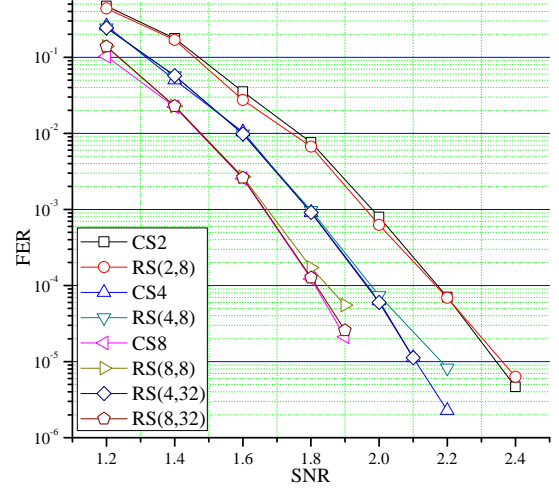


Fig. 5. FER performance simulation for an (8192, 4096) polar code

Based on the simulation results, the following conclusions are made:

- The performance of the proposed RS algorithm is affected by the list size L . For the (8192, 4096) polar code, the FER performances of RS(2, 8) is close that of CS2. However, RS(4, 8) and RS(8, 8) show performance degradation when the FER is blow 10^{-4} .
- In order to achieve good error correction performance, for the proposed RS algorithm, the threshold value W_T should be large enough. A larger W_T will transfer more rate-1 nodes to T_1 nodes, which in turn increases the chance that a correct codeword shows in the final lists. For the (8192, 4096) polar codes, RS(4, 8) and RS(8, 8) perform worse than RS(4, 32) and RS(8, 32), respectively, when the SNR is large.
- The side effect of increasing W_T is that both the decoding complexity and latency will increase since more T_1 nodes are generated. Based on simulation results shown in Fig. 5, a dynamic W_T can be adopt for the proposed RS algorithm in order to achieve the most latency reduction at different SNR regions while maintaining the error correction performance.

D. Hardware implementation of the proposed SLMLD

In this paper, an efficient hardware implementation of the proposed SLMLD algorithm is shown in Fig. 6, where the

corresponding list size $L = 4$, and the architectures for other L values can be inferred. As shown in Fig. 6, the node metric generation (NMG) unit finds L minimum node metrics and their corresponding constituent codes for each decoding path. For the decoding path l , the extended path metrics PM_l^j 's are obtained by adding the node metrics with the path metric PM_l , which is stored in registers and initialized with 0. BBS_{8-4} in Fig. 6 denotes the BBS with 8 metrics to be sorted. Two stages of BBS_{8-4} find the 4 minimum extended path metrics and their corresponding constituent codes and list indices.

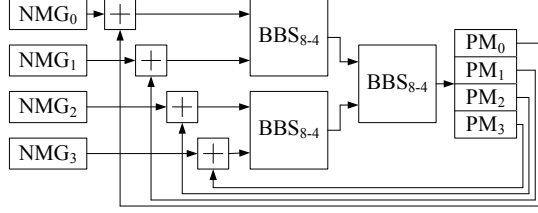


Fig. 6. The proposed architecture for SLMLD

The hardware architecture of the NMG unit is shown in Fig. 7. Since the maximum value of I_v is 8 for any T_1 node, there are at most $2^8 = 256$ candidate constituent codes for a T_1 node v . Each Enc unit in Fig. 7 is responsible for generating a candidate constituent code based on the encoding of polar codes. For $j = 0, 1, \dots, 2^{I_v} - 1$, the LLR selection unit, LS_j , and the summation unit, SUM_j , work together to compute the node metric NM_l^j shown in Eq. (8). Based on the input LLR vector $\alpha_{v,l}$, LS_j outputs an LLR vector which has the same amount of elements as that of $\alpha_{v,l}$. For $k = 0, 1, \dots, 2^{n-t_v} - 1$, the k -th output LLR is 0 only if $m_k = 0$. Otherwise, the output LLR is $|\alpha_{v,l}[k]|$. The SUM_j unit just adds up all its input LLRs sent from LS_j and outputs the corresponding node metric. The minimum two LLRs computation (MC) unit in Fig. 7 finds out the first and the second minimum LLRs and their corresponding constituent codes among all its inputs. When $L = 4$, as shown in Fig. 7, the computed node metrics are divided into 4 groups and fed to 4 MC units, respectively. The BBS_{8-4} unit generates 4 finally survived node metrics and their corresponding constituent codes.

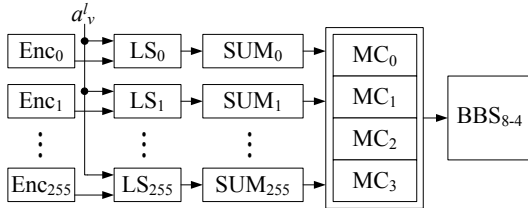


Fig. 7. Hardware architecture of the proposed NMG unit

In this paper, the proposed architecture for the SLMLD algorithm is synthesized under a TSMC 90nm CMOS technology. With 4 stages of pipeline registers, it achieves a frequency of 400MHz and consumes 1.07 million standard NAND gates.

For our implementation, when a T_1 node is activated, it will take 4 clock cycles to find the surviving constituent codes and decoding paths. The area of the architecture of the SLMLD algorithm is almost the same as an LLR based list decoder with $L = 4$.

E. Comparisons of decoding clock cycles and latency

Since the detailed decoding cycles of list decoders are related with a detailed hardware architecture, in this paper, the decoding latency comparison is performed based on the assumption that the partial parallel list architecture [10] is employed and there are $P = 128$ processing units for each decoding path. Let N_R denote the clock cycles used to decode a codeword for decoders with the proposed RS algorithm. Then $N_R = N_L + N_P$, where N_L and N_P are cycles used on the LLR computation and path pruning, respectively. Besides, $N_P = N_a N_s$, where N_a is the times that a T_1 node is activated and N_s is the number pipelines inserted in the implementation of the SLMLD algorithm. Let N_C denote the clock cycles used to decode a codeword for decoders with the CS algorithm. Then $N_C = 2N + \frac{N}{P} \log_2(\frac{N}{4P}) + N_R$ [10], where N and R are the code block length and rate, respectively.

For the aforementioned (8192, 4096) polar code used in our simulations in Section III-C, $N_L = 1207$, $N_a = 441$ and $N_s = 4$ when $W_T = 32$ and $L = 4$. Thus, $N_R = 1207 + 441 \times 4 = 2971$. Meanwhile, the cycles $N_C = 2 \times 8192 + \frac{8192}{128} \times \log_2(\frac{8192}{512}) + 4096 = 20736$. With the proposed RS decoding algorithm, the clock cycles used for decoding one codeword is reduced by about 6.97 times.

Under the UMC 90nm CMOS technology, the (8192, 4096) list polar decoder can achieve a frequency of 412MHz [12] when list size $L = 4$. Since the list decoder with the proposed RS decoding algorithm need only to change the path pruning part, the proposed list decoder can only achieve a frequency of 400MHz under 90nm technology. Thus, the decoding latency is reduced by about 6.77 times due to the proposed RS decoding algorithm when $L = 4$.

IV. CONCLUSION

In this paper, a reduced latency decoding algorithm for polar codes is proposed. The hardware implementation of the SLMLD is also discussed. The future work includes studying the performances of the proposed RLLD algorithm when FER is below 10^{-10} . Besides, more efficient implementations of the proposed SLMLD algorithm when list size is large will be investigated.

REFERENCES

- [1] E. Arkan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Info. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
- [2] E. Sasoglu, E. Teltar and E. Arkan, "Polarization for arbitrary discrete memoryless channels," in *Proc. IEEE Int. Symp. on Information Theory*, 2009, pp. 144–148.
- [3] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE Int. Symp. on Information Theory*, Jul. 2011, pp. 1–5.
- [4] I. Tal and A. Vardy, "List decoding of polar codes," in <http://arxiv.org/abs/1206.0050>.

- [5] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, Dec. 2011.
- [6] C. Zhang and K. K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Processing*, vol. 61, no. 10, pp. 2429–2441, Mar. 2013.
- [7] G. Sarkis and W. J. Gross, "Increasing the Throughput of Polar Decoders," *IEEE Commun. Lett.*, vol. 17, no. 9, pp. 725–728, Apr 2013.
- [8] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Trans. on Circuits Syst. I, Reg. Papers*, to appear.
- [9] C. Zhang and K. K. Parhi, "Latency analysis and architecture design of simplified sc polar decoders," *IEEE Trans. on Circuits Syst. II, Exp. Briefs*, vol. 61, no. 2, pp. 115–119, Feb. 2014.
- [10] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross and A. Burg, "Tree search architecture for list SC decoding of polar codes," in <http://arxiv.org/abs/1303.7127>.
- [11] J. Lin and Z. Yan, "Efficient list decoder architecture for polar codes," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, Jun. 2014, to appear.
- [12] A. Balatsoukas-Stimming, M. B. Parizi and A. Burg, "LLR-Based Successive Cancellation List Decoding of Polar Codes," in <http://arxiv.org/pdf/1401.3753v1.pdf>.
- [13] K. E. Batcher, "Sorting networks and their applications," in *Proc. ACM spring joint computer conference*, Apr. 1968, pp. 307–314.